Universitat Politècnica de Catalunya
Facultat d'Informàtica

Departament de Ciències de la Computació

Treball de final de grau:

Sistema predictiu per a evolució de pacients. Aplicació al cas d'asma infantil

---

Autor del projecte:
Ivan de la Rubia

Director del projecte:
Ricard Gavaldà

Juny 2015

Agraïments

Primer de tot voldria agrair a Ricard Gavaldà per tota l'ajuda que m'ha brindat durant el projecte, i per haver estat disponible en qualsevol moment per ajudar.

També vull agrair a l'Hospital de Sant Joan de Déu el fet d'haver-nos donat dades per realitzar els experiments, ja que aquestes dades han fet que el projecte sigui realment interessant.

De la mateixa manera, vull agrair a la meva família i als meus amics (tant de la facultat com de fora), per haver-me ajudat en tot moment.

Resum

El sistema sanitari enregistra moltes dades que després no s'exploten suficientment. Podem treure partit d'aquestes dades mitjançant tècniques de *machine learning* i *data mining* per millorar processos, diagnòstics i tractaments dels pacients.

L'Hospital de Sant Joan de Déu ha posat a la nostra disposició dades de pacients d'asma infantil, per tal que nosaltres fem un anàlisi i podem predir l'evolució de la malaltia en un pacient segons el seu estat actual, i més concretament si el pacient haurà d'anar a urgències en un temps proper.

Per tal d'aconseguir això, hem desenvolupat una aplicació que pot treballar amb dades de pacients i episodis, fer un preprocessament i realitzar entrenaments amb algorismes que han ofert resultats prometedors donada la qualitat de les dades que hem rebut. Aquest *software* és prou genèric per poder tractar altres malalties i no només l'asma infantil.

Per altra banda, hem dissenyat una aplicació de cara a ser utilitzada en un hospital, de manera que sigui realment senzill per a un metge executar aquesta aplicació i poder predir l'evolució de pacients per tal d'ajudar a prendre les seves decisions.

Resumen

El sistema sanitario guarda muchos datos que después no explota suficientemente. Podemos sacar partido de estos datos mediante técnicas de *machine learning* y *data mining* para mejorar procesos, diagnósticos y tratamientos de los pacientes.

El Hospital de Sant Joan de Déu ha puesto a nuestra disposición datos de pacientes de asma infantil, para que nosotros hagamos un análisis y podamos predecir la evolución de la enfermedad en un paciente según su estado actual, y más concretamente si el paciente tendrá que ir a urgencias próximamente.

Para conseguir esto, hemos desarrollado una aplicación que puede trabajar con datos de pacientes y episodios, hacer un preprocesamiento y realizar entrenamientos con algoritmos que han ofrecido resultados prometedores dada la calidad de los datos que hemos recibido. Este *software* es suficientemente genérico para poder tratar otras enfermedades y no solo el asma infantil.

Por otro lado, hemos diseñado una aplicación pensada para ser utilizada en un hospital, de manera que sea realmente sencillo para un médico ejecutar esta aplicación y poder predecir la evolución de pacientes con tal de ayudar a tomar sus decisiones.

Abstract

The health system stores a lot of data that doesn't explote enough. We can make the most out of this data by using machine learning and data mining techniques in order to improve diagnosis and treatments of pacients.

The Sant Joan de Déu Hospital has given us data from some child patients who suffer from Asthma, so that we can analyze this data and predict the evolution of a patient given the current state of a patient, and more precisely if the patient will have to visit the emergency room anytime soon.

To achieve this, we have developed an application that can work with data from hospitals, preprocess it and train models using algorithms that have given promising results given the quality of the data we have. This software is generic enough so that it can deal with other diseases aswell.

On the other side, we have developed another application meant to be used in a hospital, so that it is really easy for a doctor to run this application and predict the evolution of a patient in order to help with his desicions.

Continguts

Capítol 1

Introducció

En aquest capítol es farà una introducció al projecte, parlant del context, dels objectius, de la planificació, i del possible impacte del projecte

1.1 Context

En aquesta secció presentem el context del projecte, amb el seu origen, les àrees d'interès, els actors involucrats, i els principals objectius.

1.1.1 Origen del projecte

El projecte té el seu origen en una col·laboració ja en marxa entre el grup del director i l'hospital per investigar les possibilitats d'aplicar tècniques de *Data Mining* i *Machine Learning* a les dades que l'hospital ha enregistrat. Es va triar com a primer cas d'estudi l'estudi de l'asma infantil, ja que és una de les especialitats de l'hospital a nivell europeu de la qual es disposa de força dades. Després d'un primer contacte amb les dades, el director va proposar el desenvolupament d'un *software* que, a més d'aplicar-se a aquest cas d'ús, pogués ser fàcilment extensible a altres que es triessin en el futur.

1.1.2 Àrees d'interès

En aquest projecte, es poden diferenciar clarament dues àrees d'interès. La primera és l'asma infantil, que és el que intentem ajudar a millorar amb l'aplicació resultant del

projecte.
La segona són els aspectes tècnics del projecte com la creació del programa.
A continuació es farà una descripció d'aquestes dues àrees.

1.1.2.1 Asma infantil

El projecte com ja hem dit consisteix a fer un programa per predir l'evolució de l'asma infantil en nous pacients. Per tal d'aconseguir això, hem de saber què és i algunes característiques sobre aquesta malaltia.

L'asma és una malaltia crònica del sistema respiratori caracteritzada per la inflamació i l'estrenyiment de les vies respiratòries. Aquest estrenyiment causa obstrucció i per tant dificultat per passar l'aire. Quan els símptomes empitjoren, es produeix una crisi d'asma. En una crisi greu, les vies respiratòries es poden tancar tant que els òrgans vitals no reben oxigen, podent provocar fins i tot la mort.
Es calcula que hi ha 235 milions de pacients amb asma, i és la malaltia crònica més freqüent en nens.[1][2][3]

1.1.2.2 Aspectes tècnics

Per tal de desenvolupar aquest projecte, hem de tenir en compte uns quants aspectes tècnics.

Primerament, necessitem un programa que sigui capaç de treballar de forma ràpida amb les dades que rebem de l'hospital, essent així necessari fer un preprocessament de les dades abans de poder tractar-les.

Necessitem també algorismes per treballar amb aquestes dades, per tal de donar la informació pertinent al metge amb les dades que tinguem. Aquests algorismes pertanyen al camp de la informàtica anomenat *Machine Learning*.

Finalment, haurem de treballar en eines de visualització per tal de mostrar les dades resultants d'una manera senzilla.

1.1.3 Actors

El desenvolupament del projecte involucra diversos actors, llistats i explicats a continuació.

- **Desenvolupador del projecte:** El projecte només té un desenvolupador, que sóc jo. Treballaré en tot el que sigui necessari, des de la planificació fins al programa final, passant per la recerca d'informació, tests del programa, documentació...

- **Director del projecte:** El director d'aquest projecte és Ricard Gavaldà, professor de la UPC i membre del grup de recerca LARCA (*Laboratory for Relation Algorithmics, Complexity and Learning*. El seu rol és guiar al desenvolupador per tal de poder superar les dificultats que puguin aparèixer.

- **Actors beneficiats:** La idea és que aquest *software* pugui ser d'ajuda per als doctors, per tal de poder informar a les famílies dels pacients de les possibles evolucions de la malaltia, dels tractaments que podrien funcionar millor, etc.
  El projecte s'està desenvolupant amb l'ajuda de l'hospital de Sant Joan de Déu, i és per això que podem contactar a metges en cas de dubtes o suggeriments.

1.2 Objectius

El sistema sanitari està digitalitzant la pràctica totalitat de la informació que manega, però sovint aquesta informació s'explota molt poc. L'objectiu d'aquest treball és, en col·laboració amb l'Hospital de Sant Joan de Déu, elaborar un prototipus de *software* per poder fer anàlisi predictiva de l'evolució de pacients a partir dels registres electrònics de les dades que un hospital enregistra, tant categòriques com textuals, amb eines d'aprenentatge automàtic i mineria de dades.

Un cop desenvolupat aquest *software*, es posarà en pràctica amb dades de pacients infantils d'asma, i l'objectiu és que a partir de la informació del seu historial clínic, evolució de símptomes i tractaments aplicats es pugui predir l'evolució de la malaltia a mitjà termini.

Per tal d'aconseguir aquest objectiu, el *software* comptarà amb dues funcions ben distingides. La primera consistirà a fer un preprocessament de les dades de l'hospital, per tal d'aconseguir un conjunt de dades apropiat per poder aplicar els diversos algorismes.

La segona part, serà la mateixa aplicació d'aquests algorismes al conjunt de dades resultant, mitjançant un mòdul de predicció, i la visualització dels resultats d'una manera intel·ligible per qualsevol usuari.

1.3 Planificació temporal

1.3.1 Programa

La duració del projecte és aproximadament de 4 mesos i mig, des del 2 de febrer de 2015 fins al 29 de juny de 2015. A continuació es detallarà com s'emprarà aquest temps.

Cal tenir en compte però, que aquesta planificació pot canviar durant el desenvolupament del projecte.

1.3.2 Descripció de les tasques

1.3.2.1 Planificació del projecte (fita inicial)

És la primera part que treballarem del projecte, i consta de les següents parts:

1. Definició de l'abast
2. Planificació Temporal
3. Pressupost i sostenibilitat
4. Presentació preliminar
5. Context i bibliografia
6. Plec de condicions (Bloc Especialitats)
7. Document final

1.3.2.2 Preparació de l'entorn

Abans de començar a programar l'aplicació que volem fer, cal preparar l'entorn en el qual treballarem. Tot i que no és una tasca massa extensa en duració, cal esmentar-la degut a la seva importància.

Haurem de preparar totes les eines que utilitzarem segons les tecnologies que utilitzem, i també hem d'instalar LaTeX per generar els documents. Un cop haguem instalat i configurat el *software*, podem començar a treballar en el projecte en si.

1.3.2.3 Aplicació principal

Aquesta és la part més important del projecte, i és el desenvolupament del *software* que volem fer. Es pot dividir en dues fases:

1. Programació de la part de preprocessament de dades: El primer que hem de fer és una aplicació que ens serveixi per fer un preprocessament de les dades que rebem de l'hospital. En concret, podem rebre dades de diferents especialistes, i cal fer un únic document de dades pels algoritmes d'aprenentatge automàtic i de mineria de dades. A més, es vol que aquest programa serveixi no només per les dades d'asma que utilitzarem, sinó per qualsevol tipus de dades.

2. Programació dels algorismes: Un cop tinguem el programa anterior, podem començar a treballar en la programació dels algorismes, sigui utilitzant o extenent alguns que ja venen en la llibreria que utilitzem, o fent-ne de nous.

3. Programació del visualitzador: Quan s'hagin programat els algorismes, caldrà fer una finestra on mostrar els resultats d'una manera intel·ligible per tal que els metges els puguin interpretar sense dificultats.

Com es pot veure, les dependències són molt clares. Primer hem de fer el preprocessament de dades, després programar els algorismes i finalment mostrar els resultats.

1.3.2.4 Tasca final

Finalment, es comprovarà que tot funciona correctament i es prepararà el document final que inclourà tot el desenvolupament de l'aplicació i una guia d'usuari, així com la part de la fita inicial.

1.3.3 Temps aproximat

La Taula 1.1 especifica el temps que s'emprarà en cadascuna de les tasques que hem esmentat anteriorment.

Tasca	**Temps (hores)**
Planificació del projecte	80
Preparació de l'entorn	5
Preprocessament de dades	125
Programació dels algorismes	175
Programació del visualitzador	75
Tasca final	40
Total	500

Taula 1.1: *Temps emprat en cada tasca*

1.3.4 Diagrama de Gantt

A la imatge anterior podem observar la planificació temporal en un diagrama de Gantt. Les unitats de duració son dies, i a cada dia s'espera poder treballar unes 6 hores, per tal de poder cumplir amb les 500 hores estimades del projecte.

1.3.5 Pla d'acció

La idea va ser seguir les tasques tal com les vam planejat al diagrama de Gantt, però evidentment van aparèixer complicacions que ens van fer desviar-nos del diagrama.

Vam decidir que en cas de quedar-nos sense temps, ens interessaria tenir la part de preprocessament, alguns algorismes (intentant que siguin els que millors resultats donin), i una part de visualització que encara que no sigui perfecta, amb unes petites explicacions pugui ser suficient.

Es van fer reunions amb el director del projecte cada cop que es va acabar una tasca, així com d'altres en cas quan van aparèixer dificultats per tal de decidir la millor manera de solucionar-les. Les hores necessàries que hem calculat a la taula 1.1 són suficients per acabar el projecte en el temps del qual disposem.

També es vol fer un petit informe pels metges on s'indiqui quins camps són els més importants per poder fer prediccions amb una bona precisió.

1.4 Recursos

Per a poder realitzar aquest projecte, farem ús de distintes eines de *hardware* i *software*, així com recursos humans.

1.4.1 Hardware

- PC (Intel i5 4670k, 8GB RAM, nVidia GeForce GTX 660)

1.4.2 Software

- Windows 8.1
- Sublime Text
- LaTeX
- Weka Explorer
- TortoiseSVN
- git
- KNIME

1.4.3 Recursos humans

Per a la realització d'aquest projecte, també farà falta treballar de diferents rols:

1. Director de projecte
2. Dissenyador de *software*
3. Programador
4. *Software tester*

1.5 Pressupost

Aquesta secció tracta sobre el pressupost del projecte. Per tant, conté una descripció detallada dels costos del projecte (tant materials com humans), i una anàlisi de com els diferents obstacles podrien afectar el pressupost. El pressupost es pot dividir en 4 parts que detallarem a continuació:

1. Costos associats al hardware
2. Costos associats al *software*
3. Recursos humans
4. Despeses indirectes

A més, es farà un càlcul de l'amortització de cada producte basant-nos en el seu temps de vida útil i la llargada del projecte (uns 6 mesos)

1.5.1 Costos associats al hardware

Com que el projecte només consisteix a desenvolupar un *software*, no ens cal res més que un ordinador. A la Taula 1.2 podem veure el cost associat al hardware.

Producte	**Preu**	**Vida útil**	**Amortització**
Ordinador (amb els dispositius necessaris)	1.200€	4 anys	150€
Total	1.200€		150€

Taula 1.2: *Costos associats al hardware*

1.5.2 Costos associats al *software*

En el cas del nostre projecte, els costos associats al *software* seràn molt pocs, ja que treballarem totalment amb tecnologies *open-source*. L'únic cost és l'editor de text que utilitzarem, i el sistema operatiu *Windows* per fer proves (ja que el *software* estarà disponible per *Windows*). La Taula 1.3 mostra aquestes despeses.

Producte	**Preu**	**Vida útil**	**Amortització**
Windows 8.1 Professional	119€	3 anys	19,83€
Sublime Text 3	63€	3 anys	10,5€
Weka	0€	3 anys	0€
nw.js	0€	3 anys	0€
KNIME	0€	3 anys	0€
TortoiseSVN	0€	3 anys	0€
git	0€	3 anys	0€
LaTeX	0€	3 anys	0€
Total	182€		30,33€

Taula 1.3: *Costos associats al software*

1.5.3 Recursos humans

La Taula 1.4 mostra els recursos humans necessaris per al desenvolupament del projecte.

El rol de director de projecte només treballarà en la planificació del projecte.

La tasca del dissenyador de *software* serà programar tota la part visual, fer una interfície que tingui molt en compte l'experiència d'usuari, fer la part de visualització de resultats i els tests de viabilitat.

El programador s'encarregarà de programar la part del preprocessament de dades,

Rol	**Preu/hora**	**Hores**	**Cost**
Director de projecte	50€	80h	4.000€
Dissenyador de *software*	35€	160h	5.600€
Programador	35€	175h	6.125€
Software tester	20€	85h	1.700€
Total		500h	17.425€

Taula 1.4: *Recursos humans*

i els algorismes necessaris.

Finalment, el *software tester* s'encarregarà de fer proves exhaustives per tal que tot el projecte funcioni correctament i sense errors.

1.5.4 Despeses indirectes

A la Taula 1.5 podem veure les despeses indirectes del projecte.

Producte	**Preu**	**Unitats**	**Cost aproximat**
Electricitat	0,11€/kWh	600kWh	66€
Paper	5,28€/pack	1 pack (500 fulls)	5,24€
Total			71,24€

Taula 1.5: *Despeses indirectes*

En el cas de l'electricitat, és un cost aproximat, ja que les noves tarifes d'electricitat varien per hora.

1.5.5 Control de gestió

Com hem explicat anteriorment, el pressupost podria sofrir modificacions si durant el projecte no podem seguir el pla establert.

Respecte al *hardware*, és molt improbable que necessitem més del que ja hem especificat a la secció de costos associats al hardware. Quant al *software*, és possible que

necessitem més eines que les que hem dit, però avui en dia es poden trobar alternatives gratuites a pràcticament qualsevol programa.

És per això que l'únic imprevist que ens podem trobar és relacionat amb els recursos humans, i més concretament amb les hores que dediquem al projecte.
És possible que necessitem més hores que les que hem previst anteriorment. Suposem que cada rol pot utilitzar un 10% més de les hores previstes. A partir d'aquí, reservem una part del pressupost per a aquestes possibles hores "extra", que es pot veure a la Taula 1.6

Rol	**Preu/hora**	**Hores**	**Cost**
Director de projecte	50€	8h	400€
Dissenyador de *software*	35€	16h	560€
Programador	35€	17h	595€
Software tester	20€	8h	160€
Total		49h	1.715€

Taula 1.6: *Contingències*

1.5.6 Pressupost total

A la Taula 1.7 hem fet la suma de tots els gastos associats al projecte per veure el pressupost total necessari:

Despeses	**Cost**	**Amortització**
Costos de hardware	1.200€	150€
Costos de *software*	182€	30,33€
Recursos humans	17.425€	
Despeses indirectes	71,24€	
Contingència	1.715€	
Total	20.593,24€	180,33€

Taula 1.7: *Pressupost total*

1.6 Sostenibilitat

En aquesta secció avaluarem la sostenibilitat del nostre projecte en 3 àrees:

1. Econòmica
2. Social
3. Ambiental

1.6.1 Econòmica

Ja hem parlat dels costos del projecte. Tot i així, seria possible que en el futur el *software* necessités alguna actualització (per exemple per poder acceptar nous tipus de variables que puguin introduir als historials clínics).

El preu és prou ajustat, ja que respecte als recursos humans, seria difícil fer el projecte en menys temps (de fet, no sabem si tindrem prou temps per fer tot el que voldríem). Respecte al *software*, hem utilitzat tot *open-source* menys l'editor de text i el *Windows*. Ens podríem estalviar l'editor de text, però ens suposa una gran comoditat que repercuteix en més productivitat, i per tant, menys costos. Finalment, respecte al *hardware*, podríem tenir un ordinador menys potent o amb pantalles de menys qualitat, però de nou, el fet de treballar per exemple amb dues pantalles augmenta significativament la productivitat, reduint així els costos.

1.6.2 Social

El *software* que estem desenvolupant es pot utilitzar per millorar la vida de pacients amb asma, ja que a partir de les prediccions que faci el *software*, el metge pot donar un tractament més adequat, canviar de tractament, o preveure abans de temps possibles complicacions.

També podríem dir que millora la vida dels metges, ja que els ajuda a eliminar possibles complicacions del pacient fent així la seva feina més "fàcil".

1.6.3 Ambiental

Finalment, parlem de la sostenibilitat ambiental, on el nostre projecte també pot ajudar. Els medicaments son difícils de reciclar, i és possible que utilitzant el nostre *software* es puguin donar tractaments més adients als pacients i d'aquesta manera evitar que es llencin medicaments.

Pel que fa al desenvolupament, només utilitzarem un ordinador. Per això, l'únic element que no és respectuós amb el medi ambient és l'energia que consumeix l'ordinador en tenir-lo engegat.

Capítol 2

Tècniques aplicables

2.1 Estructura de les dades

De cara a desenvolupar el *software*, farem unes suposicions generals sobre les dades que ens proporcionaria una institució mèdica per a l'estudi o predicció d'alguna malaltia. Aquestes suposicions es deriven de les dades sobre asma infantil que ja ens ha proporcionat l'Hospital de Sant Joan de Déu i de les converses amb els professionals que ens les han proporcionat. A la Secció 5.1 i l'Apèndix 6.2.4 descriurem les dades d'aquest cas concret, però cal tenir en compte que l'objectiu de fer el *software* és que sigui fàcilment adaptable a altres estudis de predicció a partir de dades mèdiques, i per això volem que sigui el màxim de genèric possible.

Suposem que les dades que rebrem consistiran en diverses taules, que poden estar representades com a full de càlcul, text separat per comes, etc. Algunes de les taules representaran esdeveniments lligats a pacients, i llavors cada fila contindrà alguna columna que representarà de manera única (encara que probablement "anonimitzada") el pacient, per poder combinar les diferents taules per pacients. Poden haver-hi altres taules auxiliars que descriuen malalties, medicaments, zones geogràfiques, unitats dins de l'hospital, proves mèdiques, o qualsevol altra cosa.

Les dades que rebrem seran principalment de tipus categòric, de text i numèriques. Una dada categòrica és una dada que té un nombre determinat de possibles valors, un exemple de dada categòrica podria ser "com se sent el nen?" i els possibles valors son "be", "regular", "malament". Per altra banda, les dades de tipus text seran anotacions en text lliure que haurà escrit el metge a l'hora de fer l'informe, i les haurem de processar prèviament, amb tècniques de *text mining* com poden ser extracció de paraules clau. Finalment, les dades de tipus numèriques no necessiten gaire processament previ. Un exemple de dada numèrica podria ser l'edat del pacient en el moment de la consulta.

Un problema potencial és que molts dels camps poden estar en blanc perquè el metge que atén el nen no ha entrat cap valor, si el programa no fa obligatòria l'entrada. En el cas de l'asma, aquest és un cas molt freqüent, i a més un camp en blanc d'una pregunta sí/no no sempre pot interpretar-se com a "no". Això serà en molts casos un problema per fer una predicció acurada perquè és informació rellevant però desconeguda.

2.2 Tècniques aplicables

L'objectiu d'aquesta secció és donar informació sobre les tècniques que podem aplicar en el nostre projecte.

2.2.1 *Machine learning*

El *machine learning* és un camp de la intel·ligència artificial que estudia els algorismes que poden aprendre a partir de dades. Aquests algorismes, creen uns models que després poden aplicar per tal de fer prediccions o decisions.

És un tema molt tractat a la literatura, del qual existeixen molts llibres i publicacions en tots els àmbits de la vida (medicina, buscadors d'Internet, visió per computador...)

La tria dels algorismes a usar ha d'equilibrar diversos factors: la seva capacitat de predir amb alt encert, els recursos computacionals que usen, i la interpretabilitat dels models per part dels experts humans (especialment important en el cas dels metges).

Existeixen diferents tipus d'algorismes de *machine learning* per a classificar, com poden ser aprenentatge a partir d'arbres de decisió, xarxes bayesianes, SVM (*support vector machines*), etc.[4]

2.2.2 *Text mining*

El *text mining* es podria definir com el procés d'extreure informació a partir d'un text. [5] En el camp de la medicina, és un procés molt estès i sobre el qual hi ha diversos estudis en els quals ens podem basar en cas que sigui necessari.[6][7][8]

En el nostre cas s'aplicarà perquè les dades d'una història clínica hospitalària típicament contenen dos tipus de dades: camps estructurats, on el metge ha de triar

d'una sèrie d'opcions predefinides o entrar un valor numèric, i dades no estructurades (imatges, text, etc) i principalment camps de text lliure on escriu en català o castellà. Aquests camps lliures contenen molta informació clínica però són més difícils de tractar.

Tot i així, en el nostre projecte tindrà una importància relativament petita, ja que ens volem centrar més en la part de predicció, i en el cas de necessitar extreure informació d'un text, s'utilitzarien tècniques senzilles com per exemple la cerca de paraules clau a un text (havent definit prèviament aquestes paraules clau).

2.2.3 Tecnologies web

Hem decidit treballar amb tecnologies web, ja que actualment estan en auge i tenen tot el potencial que les tecnologies d'escriptori clàssiques. A més, gràcies a aquestes tecnologies web podrem fer tant aplicacions d'escriptori com aplicacions per al mòbil sense haver de canviar pràcticament el codi.

2.3 Algorismes d'aprenentatge

A continuació s'expliquen els tres algorismes que hem escollit pel nostre *software*. No es vol entrar en detall en l'explicació, que podria ser molt extensa per cada algorisme, si no que es vol que s'entengui la idea bàsica darrere de cadascun, i perquè pot anar be pel nostre programa.

2.3.1 Arbres de decisió: C4.5

L'algorisme C4.5 [9] és un algorisme utilitzat per generar arbres de decisió. És molt similar a un algorisme anomenat ID3 que va desenvolupar previament Quinlan, però amb algunes millores, com per exemple que suporta *missing values*.

La principal diferencia amb altres algorismes generadors d'arbres de decisió es que el criteri que utilitza per dividir es basa en la teoria de la informació.

L'algorisme C4.5 utilitza una mètrica anomenada "guany d'informació" per decidir quines divisions fer a l'arbre. El guany d'informació és la diferència entre l'entropia[1] del conjunt de dades després i abans que ha estat dividit segons els valors de l'atribut.

[1]Entropia de Shannon, al camp de la teoria de la informació

La intuïció és que d'aquesta manera es busca l'atribut que més ordre introdueix en el conjunt de dades, i per tant que millor l'explica.

Un cop es té sobre quin atribut es vol fer la divisió, la generació de l'arbre és trivial:

1: **function** GenerarArbreDeDecisio(dataset)
2: Comprovar casos base
3: millorAtribut $\leftarrow$ $(a \in atributs \mid gain\_ratio(a)$ *és màxim*$)$
4: Crear un node que faci una divisió en millorAtribut
5: **for all** Subllista que s'ha creat al fer la divisió **do**
6: GenerarArbreDeDecisió per aquesta subllista, i posar-lo com a fill del node
7: **end for**
8: **end function**

Els cas base de la recursivitat és si el *dataset* que rep la funció és massa simple, crea una fulla amb la classe majoritària al dataset. Que un *dataset* sigui massa simple pot voler dir, per exemple:

- Que pràcticament totes les entrades del conjunt tenen la mateixa classe.
- No s'ha trobat cap atribut que faci una bona divisió, que vol dir que la classe sembla "soroll".
- El conjunt es massa petit i no val la pena seguir dividint.

Això es pot controlar mitjançant alguns paràmetres. A la implementació que farem servir per al projecte (J48, de la llibreria Weka), un paràmetre és la mida mínima del conjunt perquè es consideri "no simple". Mitjançant aquest valor es poden obtenir arbres més grans o més petits, doncs podem decidir a partir de quan no fer més divisions sino fer fulles.

L'algorisme C4.5 és més complex que això, ja que després poda l'arbre, tracta els *missing values* de manera especial, etc, però amb aquesta introducció podem tenir una idea clara de com treballa.

Els principals motius per triar aquest algorisme per al nostre programa són dos. Un que és ràpid i pot tractar amb conjunts de dades grans. El segon, potser més important, és que els arbres de decisió obtinguts són relativament interpretables. Cada camí de l'arrel a una fulla pot veure's com una "tipologia de pacients", i els atributs que es consulten pel camí com una descripció de què tenen en comú i què els diferencia d'altres tipologies. És clar, això és més o menys cert segons la mida de l'arbre: un arbre amb 1000 fulles difícilment podrà entendre's bé.

2.3.2 Random forest

La idea bàsica dels *random forests* [10] és generar molts arbres de decisió utilitzant un nombre petit $m << M$ de totes les variables disponibles M. Un cop és té el conjunt d'arbres (bosc) i es vol classificar una entrada, es decideix la classificació a cada arbre, i la que tingui més "vots" serà la classificació d'aquesta entrada. Hem escollit aquest algorisme degut a alguns dels seus avantatges [11]:

- És un dels algorismes que millor precisió ofereix
- És molt ràpid per a conjunts de dades grans
- Té en compte totes les variables, no ignora cap com fan altres algorismes
- Manté la precisió tot i que falti una gran quantitat de dades (*missing values*)

Com a inconvenients, cal triar diversos paràmetres com ara nombre d'arbres i atributs a triar a cada arbre, i a més el model resultant, la votació d'uns quants arbres, és difícil d'interpretar per part dels experts humans, tant en general com en el resultat que dóna amb una instància en concret.

2.3.3 Naive Bayes

L'últim algorisme que utilitzarem en el nostre *software* és el Naive Bayes. És un algorisme basat en el teorema de Bayes, que diu:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

on $P(A)$ és una probabilitat a priori, $P(B|A)$ és la probabilitat condicionada de B donat A, i $P(A|B)$ és la probabilitat a posteriori.

Aquesta fòrmula només ens permet calcular la probabilitat d'A donada només una variable B. Aquí és on entra el *naive*[2] de l'algorisme. L'algorisme de *naive Bayes* suposa que totes les variables de l'entrada són independents, per tant la probabilitat que una entrada N amb n variables sigui de la classe C_j es calcula simplement multiplicant [4]:

$$P(C_j|N) = \frac{P(N_1|C_j) \cdot ... \cdot P(N_n|C_j) \cdot P(C_j)}{P(N_1) \cdot ... \cdot P(N_n)}$$

[2]naive en anglès vol dir ingenu

Com que volem fer aquest càlcul per cada classe, el denominador no canvia i el podem obviar, i simplement comparar el numerador. La classe que tingui el numerador més gran serà la seleccionada per classificar aquesta entrada, doncs serà la classe més probable donada la entrada N.

És possible veure que Naive Bayes en el fons implementa una regressió lineal: Prenent logaritmes, el producte de probabilitats es transforma en una suma de coeficients per variables. És possible doncs interpretar el valor dels coeficients com a una mesura del poder predictiu de cada variable, cosa atractiva per a aplicacions mèdiques on calgui entendre els models obtinguts.

Naive Bayes és un algorisme molt ràpid, que no necessita paràmetres per funcionar, i que és dels primers que s'utilitza en qualsevol tasca de classificació. Encara que en moltes tasques complexes d'aprenentatge no pot competir amb algorismes més complexos, és sorprenent la quantitat de contextos on és competitiu. S'ha vist que aquest és el cas en algunes aplicacions mèdiques [12] [13], i de fet ho serà en el nostre cas comparat amb els altres dos algorismes triats.

Capítol 3

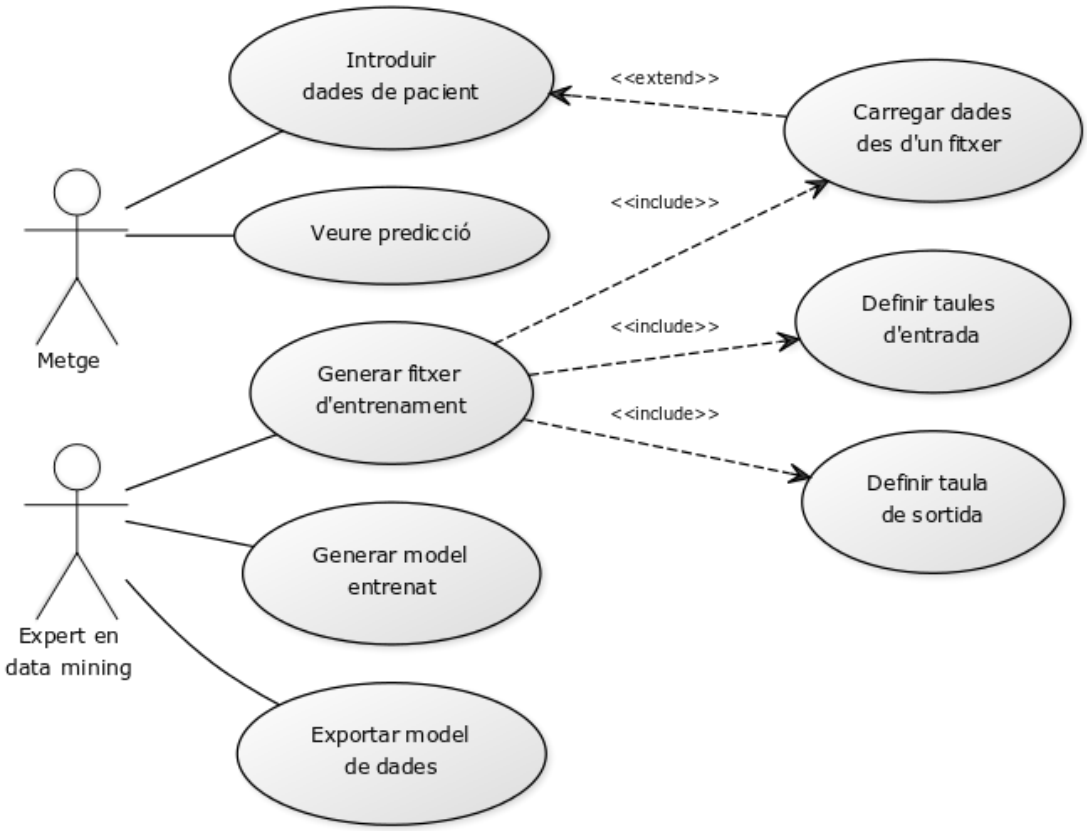
Disseny

Com ja hem explicat als objectius del projecte, la idea era fer un *software* que ens permetès fer el preprocessament de les dades, l'entrenament, i la predicció/visualització. Un cop es van fer els primers mockups (apèndix 6.2.4), es va decidir que l'aplicació seria massa complexa d'utilitzar per un metge, i seria millor separar les tasques en dues aplicacions.

Per un costat, tindríem un informàtic o expert en mineria de dades, que definiria les taules d'entrada (el format de les dades de l'hospital) i la taula de sortida (les variables sobre les quals es faria l'entrenament després). Un cop es tingués aquest model, es podríen carregar les dades de l'hospital, generant així un fitxer d'entrenament (que conté les dades de pacients, episodis, urgències, etc) que s'utilitzaria per entrenar amb el mòdul de predicció. Ara que ja tenim el model entrenat, es pot fer una aplicació molt senzilla pels metges que carregués aquest model, demanés les dades de nous pacients, i fes una trucada al *Weka* per després mostrar els resultats.

3.1 Casos d'us

A partir del raonament fet anteriorment, es va definir un diagrama de casos d'us:



Cas d'us	Introduir dades de pacient
Actors	Metge
Descripció	S'introdueixen les dades d'un o més pacients ja sigui omplint tots els camps que apareixen, o carregant les dades des d'un fitxer
Precondició	-
Excepcions	Les dades del fitxer no s'han pogut carregar
Postcondició	S'han afegit un o més pacients, dels quals es podrà veure la seva predicció

Taula 3.1: Cas d'us 1: Introduir dades de pacient

Cas d'us	Veure predicció
Actors	Metge
Descripció	Es mostren els resultats de la predicció que s'ha fet per cada pacient
Precondició	S'han afegit les dades d'algun pacient anteriorment
Excepcions	-
Postcondició	Es pot veure el resultat de la predicció per cada pacient

Taula 3.2: Cas d'us 2: Veure predicció

Cas d'us	Generar fitxer d'entrenament
Actors	Expert en mineria de dades
Descripció	Es genera un fitxer d'entrenament que té l'estructura de la taula resultant i totes les dades processades
Precondició	• S'han definit una o més taules d'entrada • S'ha definit la taula de sortida • S'han carregat les dades de les taules d'entrada
Excepcions	Hi ha algun problema amb el codi personalitzat en algun atribut de sortida
Postcondició	S'ha generat un fitxer d'entrenament a la ruta especificada, amb totes les dades processades

Taula 3.3: Cas d'us 3: Generar fitxer d'entrenament

Cas d'us	Generar model entrenat
Actors	Expert en mineria de dades
Descripció	S'entrena un model utilitzant algun algorisme disponible basat en l'arxiu d'entrenament generat anteriorment
Precondició	Cal tenir al menys un fitxer d'entrenament generat
Excepcions	-
Postcondició	S'ha generat un model entrenat que s'utilitzarà en l'aplicació del metge

Taula 3.4: Cas d'us 4: Generar model entrenat

Cas d'us	Exportar model de dades
Actors	Expert en mineria de dades
Descripció	S'exporta un model en format JSON que conté l'estructura de les taules d'entrada, de la taula de sortida, i el codi personalitzat que s'hagi escrit.
Precondició	-
Excepcions	-
Postcondició	S'ha exportat el model

Taula 3.5: Cas d'us 5: Exportar model de dades

Cas d'us	Carregar dades des d'un fitxer
Actors	Metge, Expert en mineria de dades
Descripció	Permet carregar dades de pacients, episodis, o el que sigui des d'un fitxer de dades
Precondició	-
Excepcions	-
Postcondició	S'han carregat les dades

Taula 3.6: Cas d'us 6: Carregar dades des d'un fitxer

Cas d'us	Definir taules d'entrada
Actors	Expert en mineria de dades
Descripció	Permet definir les taules d'entrada, així com els seus atributs
Precondició	-
Excepcions	-
Postcondició	S'han definit les taules d'entrada

Taula 3.7: Cas d'us 7: Definir taules d'entrada

Cas d'us	Definir taula de sortida
Actors	Expert en mineria de dades
Descripció	Permet definir els atributs de la taula de sortida
Precondició	S'ha definit alguna taula d'entrada prèviament
Excepcions	-
Postcondició	S'ha definit la taula de sortida

Taula 3.8: Cas d'us 8: Definir taula de sortida

3.2 Jerarquia de classes

El framework AngularJS ens permet programar aplicacions en Javascript utilitzant un patró molt semblant al MVC (*Model-View-Controller*) anomenat MVVM (*Model-View-ViewModel*).

La principal diferència és que els controladors la feina que fan en AngularJS és posar a disposició de les vistes les dades dels models, i viceversa, amb el que es coneix com a *2-way data binding*[1]. El que vol dir això és que si les dades canvien en el model, la vista s'actualitza automàticament a l'instant, i si és canvia res a la vista, el model s'actualitza automàticament també.

A continuació es descriuren els models (implementats com a serveis[2]), les vistes i els controladors:

3.2.1 Models

- TableService: És el model que guarda les taules d'entrada i la de sortida, així com els seus atributs. Té funcions per afegir i esborrar taules, seleccionar les claus primàries, afegir i esborrar atributs...
- DataService: Simplement s'encarrega d'emmagatzemar les dades carregades des dels fitxers en memòria, i permet l'accés de diferents controladors a aquestes dades.

3.2.2 Controladors (*view-models*)

- MenuCtrl: És el controlador que s'encarrega de manegar en quin punt de l'aplicació estem en cada moment.
- InputCtrl: Gestiona tot el que està relacionat amb la definició de les taules d'entrada.
- OutputCtrl: Gestiona tot el que està relacionat amb la definició de la taula de sortida.
- LoadDataCtrl: S'encarrega de carregar les dades des d'arxius, i emmagatzemar-les en les estructures de dades explicades en detall a la Secció 3.3.3.

[1] https://docs.angularjs.org/guide/databinding
[2] https://docs.angularjs.org/guide/services

- ExportCtrl: Aquest últim controlador, s'encarrega de generar la taula resultant, i d'exportar els models entrenats. En aquesta mateixa classe estan definits els algorismes que el nostre *software* pot utilitzar, així com els seus paràmetres. Per tant, afegir o treure els algorismes es simplement definir-los aquí, i definir la funció que faci la crida al *Weka* amb els paràmetres adients.

3.2.3 Vistes

- Index: La vista principal, conté el menú, i una "sub-vista", que depèn de la pantalla a la que ens trobem.
- EditInputTable: Mostra els camps necessaris per editar la taula d'entrada sel·leccionada.
- EditOutputTable: Mostra els camps necessaris per editar la taula de sortida.
- LoadData: Apareixen les taules d'entrada que hem definit, i la opció per carregar les dades d'aquestes taules.
- ExportData: Permet generar la taula resultant, així com fer els diferents entrenaments i veure els resultats.

3.3 Mòdul d'anàlisi

En aquesta secció s'explicarà la implementació de les diferents funcions del *software*. Cal destacar que durant l'explicació utilitzaré indisintament objecte per referir-me a un diccionari, i propietat d'un objecte per referir-me a una entrada d'un diccionari, ja que els objectes en Javascript es comporten com diccionaris a la resta de llenguatges.

3.3.1 Definició de taules d'entrada

L'estructura utilitzada per guardar les taules d'entrada era simplement una llista de taules, on cada taula conté 3 propietats:

- **Nom de la taula**: Per defecte, el nom de la taula és "Taula " i el número total de taules + 1.
- **Atributs**: Una llista (per defecte buida) dels atributs que formen aquesta taula.

- **Clau primària**: Guarda l'identificador de l'atribut que és la clau primaria de la taula.

Pel que fa als atributs de les taules d'entrada, simplement eren objectes amb dues propietats:

- **Nom de l'atribut**: Per defecte, el nom dels atributs és "Atribut " i la quantitat d'atributs de la taula + 1.
- **Tipus**: Indica el tipus d'aquest atribut.

3.3.2 Definició de taula de sortida

En el cas de la taula de sortida, només utilitzem un únic objecte amb 3 propietats:

- **Pre-codi**: Permet definir un codi que s'executarà abans de calcular els valors de cada atribut al generar la taula resultant.
- **Post-codi**: Permet definir un codi un cop s'han calculat tots els valors dels atributs.
- **Atributs**: Igual que en les taules d'entrada, és una llista que guarda tots els atributs.

En la taula de sortida, els atributs són més complexos:

- **Tipus**: Defineix el tipus d'atribut. Pot ser "Join" o "Key-value". En la documentació del *software* està explicada la diferència entre els dos tipus.
- **Nom**: El nom de l'atribut.
- **Original**: Conté una copia de l'atribut del qual sortirà el seu valor. En el cas de ser del tipus "key-value", és l'atribut que s'utilitzarà com a key.
- **Funció**: Conté la funció que s'executarà sobre aquest atribut. En el cas de ser del tipus "key-value", és la funció que decidirà si l'atribut es key o no.
- **Codi personalitzat**: Permet definir un codi en comptes d'una funció predefinida.
- **Keyword**: Algunes funcions necessiten un camp adicional, per exemple les funcions que comproven si un atribut conté una paraula. Aquesta paraula es guarda en la propietat *keyword*.

- **Eliminar**: Si es posa aquesta propietat a true, aquest atribut no sortirà a l'arxiu final amb el que es farà l'entrenament.

Els atributs que són del tipus "key-value", tenen a més 4 propietats addicionals que són iguals que "Original", "Funció", "Codi personalitzat" i "Keyword", però afecten a l'atribut del *value*,

3.3.3 Càrrega de les dades

Per carregar les dades, com ja hem dit anteriorment, utilitzem la llibreria Papaparse. Un cop s'ha llegit el fitxer, les dades es guarden en memòria utilitzant una estructura molt concreta.

L'estructura utilitzada per guardar les dades és una mica peculiar, però ens serveix per després poder calcular tots els valors resultants de manera eficient, i fins i tot, paralel·la.

Per cada taula, es crea un diccionari. Aquest diccionari té tantes entrades com claus primaries diferents hi hagi en les dades que carreguem. Cada entrada d'aquest diccionari, alhora, conté un altre diccionari, amb tantes entrades com atributs d'entrada hi hagin en el nostre model. Finalment, dins d'aquest últim diccionari, tenim una llista que conté tots els valors d'aquest atribut per cada clau primaria.

Posem un exemple per visualitzar-ho millor. Suposem que les dades que volem carregar son les següents:

Codi pacient (clau primaria)	**Simptomes de rinoconjuntivitis**
1000	Si
1000	No
1001	Si
1001	Si

Taula 3.9: Dades d'exemple

Primerament, tindríem un diccionari amb tantes entrades com claus primaries tinguin les dades d'entrada, en aquest cas 2, i cada diccionari conté alhora un altre diccionari:

```
{
    1000: {},
    1001: {}
}
```

Aquests diccionaris contenen tantes entrades com atributs té la taula d'entrada. i cadascuna d'aquestes entrades té una llista amb tots els valors d'aquests atributs:

```
{
    1000: {
        "codi_pacient": [1000, 1000],
        "simptomes_rinoconjuntivitis": ["Si", "No"]
    },
    1001: {
        "codi_pacient": [1001, 1001],
        "simptomes_rinoconjuntivitis": ["Si", "Si"]
    }
}
```

El cost d'accés a una propietat d'un objecte depèn del motor de Javascript que s'utilitzi. En el nostre cas, utilitzem io.js que està basat en el motor V8 de Chrome. Aquest motor afirma tenir una implementació més ràpida que les *hash tables*[3]

3.3.4 Entrenament

Un cop tenim les dades en la nostre estructura, el procés per calcular la taula resultant és el següent:

```
1: function GenerarTaulaFinal(data, atributs_sortida)
2:     result ← {}
3:     for all atribut ∈ atributs_sortida do
4:         for all clau_primaria ∈ data[atribut.taula] do
5:             if ∄ result[clau_primaria] then
6:                 result[clau_primaria] ← {}
7:             end if
8:             result[clau_primaria][atribut.nom] ← resultat()
9:         end for
10:    end for
11:    return result
12: end function
```

En executar aquest algorisme, suposem que ja tenim les dades carregades a la variable data. El que fa l'algorisme és, per cada atribut de sortida i per cada clau primaria, calcular el valor resultant d'aquest atribut per a aquesta clau. El cost d'aquest algorisme depèn de la funció que s'utilitzi per calcular cada atribut, però com que les funcions que hem definit per defecte poden tenir cost O(1) o lineal (depenèn de la

[3]https://developers.google.com/v8/design#fast-property-access

funció que triem per l'atribut), suposarem que el cas pitjor és cost lineal. Sigui n la quantitat d'atributs de sortida, m el nombre de taules, i $|m_i|$ la quantitat de dades que té la taula i-éssima, definim k com la quantitat màxima de dades d'entre totes les taules.

$$k = \max_{1 \leq i \leq m} |m_i|$$

El cost del nostre algorisme en el cas pitjor és $O(n \cdot k)$ ja que per cada atribut de sortida, es processaran, com a molt, k dades amb una funció de cost lineal.

3.4 Mòdul clínic

El programa que faran servir els metges es bastant simple. Aquest programa, únicament carregarà un model que s'extreu de l'aplicació anterior en format JSON, i demanarà al metge la informació de les variables de la taula de sortida.

Quan es té aquesta informació, es fa una crida al mòdul de predicció amb aquestes dades i el model entrenat que hem generat amb l'altre *software* i mostra al metge el resultat de la classificació.

Capítol 4

Desenvolupament i implementació

4.1 Tecnologies utilitzades

A continuació es descriuen les tecnologies i llibreries utilitzades en la creació del *software*.

- **Weka**: Weka és un conjunt *open-source* d'algorismes de *Machine Learning* desenvolupat pel group de *Machine Learning* de la Universitat de Waikato (Nova Zelanda) i extés per una gran comunitat d'usuaris. Es pot utilitzar a través d'una inferfície gràfica que incorpora, o utilitzant la línia de comandes de tal manera que es pot integrar en un altre *software* (com és el nostre cas). Està escrit en Java i és fàcilment modificable en cas de ser necessari.

  Hem fet servir aquesta llibreria per implementar el que hem anomenat "mòdul de prediccions" als capítols anteriors. Teníem prop de 50 algorismes de predicció implementats a Weka, dels quals vam triar els tres que hem explicat a la Secció 2.3. [`http://www.cs.waikato.ac.nz/ml/weka/`]

- **AngularJS**: És un *framework* mantingut per Google per crear aplicacions web dinàmiques. L'hem utilitzat com a *framework* principal sobre el qual hem desenvolupat les dues aplicacions [`https://angularjs.org`]

- **Bootstrap 3**: És un *framework* pel front-end per facilitar i fer més ràpid el desenvolupament. L'hem fet servir per tenir una interfície prou bona ràpidament, gràcies a la quantitat de classes de CSS que ja venen definides. [`http://getbootstrap.com`]

- **nw.js**: Permet fer aplicacions d'escriptori utilitzant tecnologies web, i permet fer crides a mòduls d'io.js. L'hem utilitzat per tal d'executar la nostra aplicació en

un entorn d'escriptori. [`http://nwjs.io`]

- **io.js**: És una plataforma de Javascript construida sobre el *runtime* V8 de Chrome. És un *fork* de Node.js. Com ja hem dit, nw.js permet executar mòduls d'io.js. En concret, hem fet servir el mòdul *file* per tal de carregar i guardar fitxers. [`https://iojs.org`]

- **Papaparse**: És una llibreria que ens permet parsejar arxius CSV de manera senzilla i fiable. S'ha utilitzat per carregar les dades a les dues aplicacions. [`http://papaparse.com`]

- **D3.js**: És una llibreria per manipular documents basats en dades utilitzant HTML, SVG i CSS. En el nostre cas l'utilitzem per representar els arbres que obtenim als entrenaments. [`http://d3js.org`]

Capítol 5

Aplicació al cas d'asma infantil

5.1 Dades disponibles

L'Hospital de Sant Joan de Déu ens va proporcionar dades de pacients d'asma infantil. Més concretament, ens va proporcionar dues bases de dades que contenien la següent informació:

- **Episodis**: Contenia dades de 13425 episodis, de 7294 pacients diferents. La mitja d'edat d'aquests pacients en el moment dels episodis era d'aproximadament 10 anys i mig. D'aquesta taula d'informació és d'on hem tret la majoria de les variables utilitzades per a la predicció.
- **Urgències**: Contenia informació de 4261 visites a urgències a l'hospital de Sant Joan de Déu. Com ja hem explicat anteriorment, no són totes les urgències dels nostres pacients ja que és possible que anessin a altres hospitals.

Utilitzant el nostre *software*, vam combinar aquestes dues taules en una única, on cada fila era un únic episodi d'un pacient. Aquestes files tenien 64 variables, algunes de les quals són extretes directament de les variables codificades en les taules originals, altres obtingudes amb les tècniques bàsiques de *text mining* ja explicades (extracció de paraules clau). El llistat de variables usades en aquest cas pot trobar-se en l'Apèndix B.

L'última variable, la que volem aprendre a predir, era si el pacient havia anat a urgències un temps després del episodi. D'aquesta manera, donades les dades d'un pacient nou, es podrà predir si el pacient anirà a urgències en un temps proper, i en cas de ser afirmatiu, es podrà incrementar la vigilància, adaptar o canviar el tractament

emprat, etc. Aquesta variable s'obtenia buscant en la taula de urgències el pacient de l'episodi, així com en alguns atributs de la taula d'episodis.

Per tant, la nostra variable classificadora divideix el conjunt de dades en dues classes (d'ara en endevant, classe A i B):

- El pacient **no** ha anat a urgències (a l'hospital de Sant Joan de Déu) en un temps proper a l'episodi (Classe A, 12431 episodis)
- El pacient **si** ha anat a urgències (a l'hospital de Sant Joan de Déu) en un temps proper a l'episodi (Classe B, 994 episodis)

Notem que, amb les dades que tenim, un problema important és que tenim *falsos negatius*, és a dir pacients que haurien de tenir classe positiva (perquè han hagut d'anar d'urgències) però que els tenim etiquetats com a negatius (perquè quan es va produir la urgència van anar a un hospital diferent de Sant Joan de Déu, i per tant no la tenim enregistrada com a tal). És més, no sols no sabem quins són aquests falsos negatius sinó que no sabem si són un 1%, un 10% o un 30% dels negatius que tenim.

Per a tots els experiments, es va fer la validació amb *10-folds cross validation*, és a dir, tot el conjunt de dades es partia en 10 subconjunts, s'agafava un d'ells i s'utilitzava per la validació, mentre que els 9 restants eren utilitzats per l'entrenament. Es repeteix aquest procés 10 vegades, utilitzant cada cop un subconjunt diferent per la validació. Finalment, es fa la mitja de tots els resultats per obtenir una estimació més precisa.

5.2 Conceptes previs

Abans de començar a comentar els resultats que hem obtingut, introduirem uns conceptes necessaris per a interpretar correctament els resultats.

Precisió: La precisió es defineix com el rati de les entrades ben classificades a una classe entre el total d'entrades que hem dit que són d'aquesta classe.

Recall: El recall, per altra banda, és el rati d'entrades ben classificades, entre totes les que realment són d'aquesta classe.

Taxa d'encert: La taxa d'encert és el rati d'entrades ben classificades entre el total d'entrades.

Matriu de confusió: La matriu de confusió és una taula que ens permet veure de manera ràpida la qualitat del nostre algorisme. Cada columna de la taula representa la

quantitat d'entrades en una classe predita, i les files representen la quantitat d'entrades que són realment d'aquesta classe.

Per entendre millor aquests conceptes, posem un exemple. Suposem que tenim 10000 episodis, dels quals en 2000 el pacient ha anat a urgències en un text proper. Executem un algorisme de *Machine Learning* i obtenim la següent matriu de confusió:

	A	B
A	7000	1000
B	500	1500

Taula 5.1: Exemple de matriu de confusió

Es pot veure que, efectivament, la fila A suma 8000 i la fila B suma 2000. El que veiem en aquesta taula és que dels 8000 pacients que no van anar a urgències, el nostre algorisme prediu bé 7000 i falla 1000 (diu que aniran a urgències quan no és així). Per altra banda, dels 2000 que si van a urgències, classifica be 1500 i falla 500. Finalment, també podem calcular a partir d'aquesta matriu la taxa d'encert fent la suma de la diagonal i dividint per la suma de tota la matriu.

$$taxa\ d'encert = \frac{7000 + 1500}{7000 + 1000 + 500 + 1500} = 0.85$$

Ara podem calcular la precisió i el recall per cada classe. En el cas de la classe A, la precisió es calcula com

$$precisio = \frac{7000}{7000 + 500} = 0.933$$

I el recall

$$recall = \frac{7000}{7000 + 1000} = 0.875$$

Ens interessa tenir un recall alt per als nens que sí que van a urgències (per poder actuar, i potser prevenir, el número més gran de casos possibles), però sense baixar massa la precisió.

5.3 Resultats

A continuació es descriuen els resultats obtinguts utilitzant els tres algorismes que hem explicat a la Secció 2.3.

Anticipem que, entre els classificadors que veurem a continuació, en trobarem dos que ho fan força bé globalment, i en particular a predir la classe "no visita a urgències", però no ho fan gaire bé a identificar el que ens interessa, que és la classe "visita a urgències".

5.3.1 Resultats amb J48

L'arbre resultant d'executar l'algorisme J48 en primera instància era un arbre amb un únic node que sempre retornava que el pacient no hauria d'anar a urgències.

D'aquesta manera l'algorisme J48 aconseguia una taxa d'encert del 92.6%, però un recall i una precisió de 0 pels pacients que si han d'anar a urgències, i per tant és pràcticament el pitjor resultat possible que podíem obtenir. Pel que fa al recall i la precisió dels pacients que no han d'anar a urgències, aquests eren evidentment igual a 1 doncs classificava a tots en aquest grup.

En cas de deixar l'arbre sense podar (seleccionant l'opción *Unpruned tree* al fer l'entrenament), i amb un mínim de 2 instàncies per fulla els resultats milloren lleugerament, però no suficient:

	A	B
A	12350	81
B	890	104

Taula 5.2: Matriu de confusió amb J48 sense podar

La mida de l'arbre resultant ara és de 387 nodes, 194 dels quals són fulles. La taxa d'encert ha pujat al 92.8%. D'aquesta taula podem també extreure la precisió i el recall per a cada classe:

- Classe A
  - Precisió: $12350/(12350 + 890) = 0.933$
  - Recall: $12350/(12350 + 81) = 0.993$
- Classe B
  - Precisió: $104/(81 + 104) = 0.562$
  - Recall: $104/(104 + 890) = 0.105$

El recall de la classe B (el que realment ens interesa) encara és massa petit per poder ser d'utilitat.

5.3.2 Resultats amb Random Forest

Per l'execució d'aquest algorisme, vam decidir utilitzar 10 arbres pel bosc, i cada arbre utilitzaria 6 atributs aleatoris de tots els disponibles. Som conscients que un anàlisi més detallat hauria de fer un escombrat de l'espai de paràmetres més exhaustiu, però de tota manera els resultats obtinguts en aquest projecte volen ser més exploratoris i il·lustratius que finals.

Els resultats són molt similars als obtinguts en el J48, i per tant tampoc és de massa utilitat aquest algorisme per a les nostres dades:

	A	B
A	12418	13
B	967	27

Taula 5.3: Matriu de confusió amb Random Forest

- Classe A
  - Precisió: $12418/(12418 + 967) = 0.928$
  - Recall: $12418/(12418 + 13) = 0.999$
- Classe B
  - Precisió: $27/(13 + 27) = 0.675$
  - Recall: $27/(27 + 967) = 0.027$

De nou, el recall de la classe B és excesivament baix.

5.3.3 Resultats amb Naive Bayes

Finalment, vam provar a executar l'algorisme Naive Bayes sobre el nostre conjunt de dades, i vam obtenir el millor resultat de tots els algorismes provats. La taxa d'encert va baixar respecte als altres 2 (es situa en 87.4%), però el recall de la classe B va pujar significativament:

	A	B
A	11267	1164
B	525	469

Taula 5.4: Matriu de confusió Naive Bayes

- Classe A
  - Precisió: $11267/(11267 + 525) = 0.955$
  - Recall: $11267/(11267 + 1164) = 0.906$
- Classe B
  - Precisió: $469/(1164 + 469) = 0.287$
  - Recall: $469/(469 + 525) = 0.472$

Dit amb paraules més clares, el que volen dir aquests resultats és que posant una mica més de vigilància sobre aproximadament un 12% dels pacients (la quantitat de pacients que el nostre algorisme ha classificat a "ha d'anar a urgències" respecte del total de pacients), es podrien predir i potser prevenir gairebé un 50% de les visites a urgències per asma.

Tot i que no és un resultat dolent, estem segurs que es pot millorar amb algunes de les idees que es plantegen a la Secció 6.2.4.

Capítol 6

Conclusions

Creiem que hem complert els objectius establerts a l'inici del programa. A més, la fase d'experimentació ha donat resultats acceptables amb les dades que teníem. Tal i com està el programa ara mateix, ja pot ser útil per l'anàlisi d'altres malalties de manera senzilla, però tot i així encara queda treball per fer, que es descriu a continuació.

6.1 Canvis en la planificació

A causa dels canvis que han anat apareixent durant el desenvolupament del projecte, la planificació temporal s'ha vist afectada de la següent manera:

Tasca	**Temps (hores)**
Planificació del projecte	80
Preparació de l'entorn	5
Programació del creador de models	335
Programació de l'aplicació per metges	30
Tasca final	50
Total	500

Taula 6.1: *Temps emprat en cada tasca*

Si la comparem amb la planificació de la fita inicial, veiem que hem dedicat moltes més hores al mòdul d'anàlisi, ja que és la part que més genèrica s'ha de fer (les dades de diferents hospitals poden ser molt diferents) i per això ha de ser altament configurable.

És a dir, ara les tasques que abans eren "preprocessament de dades" i "programació dels algorismes" han passat a ser una sola tasca, a la que a més li dediquem 35 hores més. Per altra banda, crear el mòdul clínic (el que abans era "programació del visualitzador") es redueix a fer una aplicació que carregui el model, demani dades i faci crides al Weka per fer les prediccions.

Com que la quantitat d'hores és la mateixa, i les noves tasques que s'han de fer pertanyen al rol de programador (com abans), el pressupost no s'ha vist afectat.

6.2 Treball futur

6.2.1 Interfície

La interfície del mòdul d'anàlisi pot ser molt millorada, ja que es va començar amb un prototip, que a mesura que s'anaven afegit noves funcionalitats al programa s'anava quedant menys i menys usable. A més, per falta de temps, algunes pantalles (com la de càrrega de les dades) van quedar molt simples, i poden ser molt millorades per mostrar informació extra, com la quantitat de dades carregades, la opció d'esborrar dades en cas d'haver-nos equivocat al carregar-les...

6.2.2 Seguretat

Ja que treballem amb dades molt sensibles, es podria millorar la seguretat del programa permetent la càrrega de les dades des de servidors segurs, sense la necessitat de descarregar-les al client. D'aquesta manera, s'evita el risc de que les dades surtin de l'entorn segur.

6.2.3 Més funcions

El programa compta amb les funcions més típiques que se solen utilitzar per processar dades, però en algunes hem hagut d'escriure codi personalitzat. La idea del programa és poder crear els models visualment i de manera senzilla, minimitzant així la quantitat de codi que l'expert en dades ha d'escriure. Per tant, com a més funcions tingui disponible per seleccionar a cada atribut, millor.

6.2.4 Millores en els resultats de l'asma

Com ja hem dit anteriorment, creiem que encara hi ha un bon marge de millora en els resultats de l'asma:

- Recopilant dades d'urgències d'altres hospitals per tal de tenir totes les urgències disponibles per analitzar
- Solventant el problema dels *missing values*: Ara per ara, els metges al omplir els informes obvien molts dels camps, ja que ells ja saben si el valor d'un camp buit vol dir "si", "no", "igual que abans", etc. Però nosaltres no sabem aquesta informació, i per tant seria bo que els metges omplissin tots els camps al fer una consulta a un nen, per tal de tenir el mínim de missing values possibles.
- Millorant les tècniques de *text mining*. Actualment només utilitzem cerca de paraules clau, i això pot supossar un problema. Imaginem un text que diu "El pacient no presenta símptomes de rinoconjuntivitis". El nostre *software* marcaria la variable "rinoconjuntivitis" a 1 ja que la paraula apareix en el text, però realment hauria de ser 0 ja que el text diu que **no** té aquests símptomes.

Cal dir que els responsables de Sant Joan de Déu ja són perfectament conscients dels dos primers problemes o mancances del seu sistema d'informació. El nostre prototipus ve a confirmar-les. Aquest és un resultat habitual de projectes de *Data Mining* en organitzacions: la recomanació de recollir informació que ara no s'està recollint, o que s'està recollint de manera incorrecta.

Referències

[1] Lung National Heart and Blood Institute. What is asthma? *(http://www.nhlbi.nih.gov/health/health-topics/topics/asthma)*.

[2] Varis autors. Asma. *(http://es.wikipedia.org/wiki/Asma)*.

[3] World Health Organization. Asthma. *(http://www.who.int/respiratory/asthma/)*.

[4] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2005.

[5] Varis autors. Text mining. *(http://en.wikipedia.org/wiki/Text_mining)*.

[6] Xiaohua Zhou and Hyoil Han. Approaches to text mining for clinical medical records. 2006.

[7] Patricia Cerrito. Data and text mining the electronic medical record to improve care and to lower costs. 2006.

[8] Muneo Kushima. Text data mining of the electronic medical record of the chronic hepatitis patient. 2012.

[9] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[10] Leo Breiman. Random forests – random features. *Technical Report*, 567, 1999.

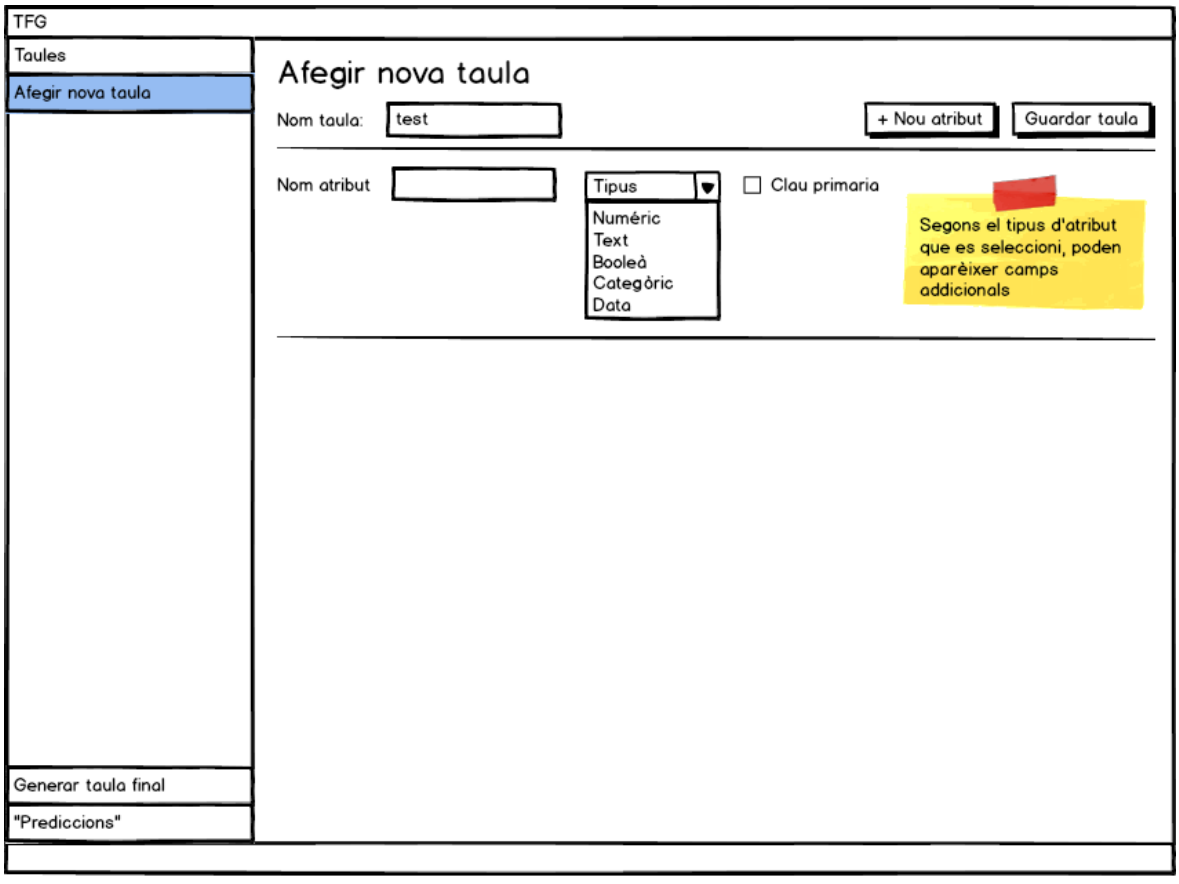
[11] Leo Breiman and Adele Cutler. Random forests - classification description. *(https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)*.

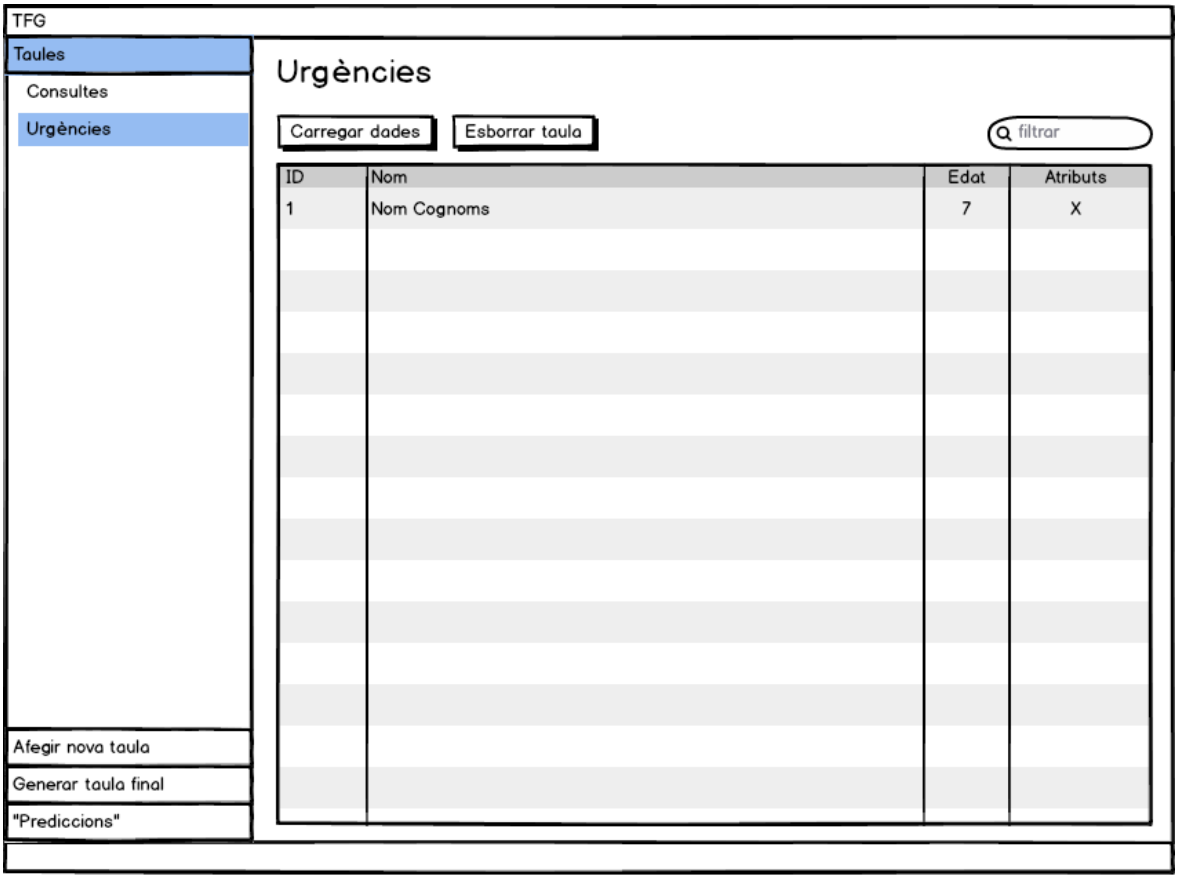
[12] Igor Kononenko. Machine learning for medical diagnosis: History, state of the art and perspective. 2001.

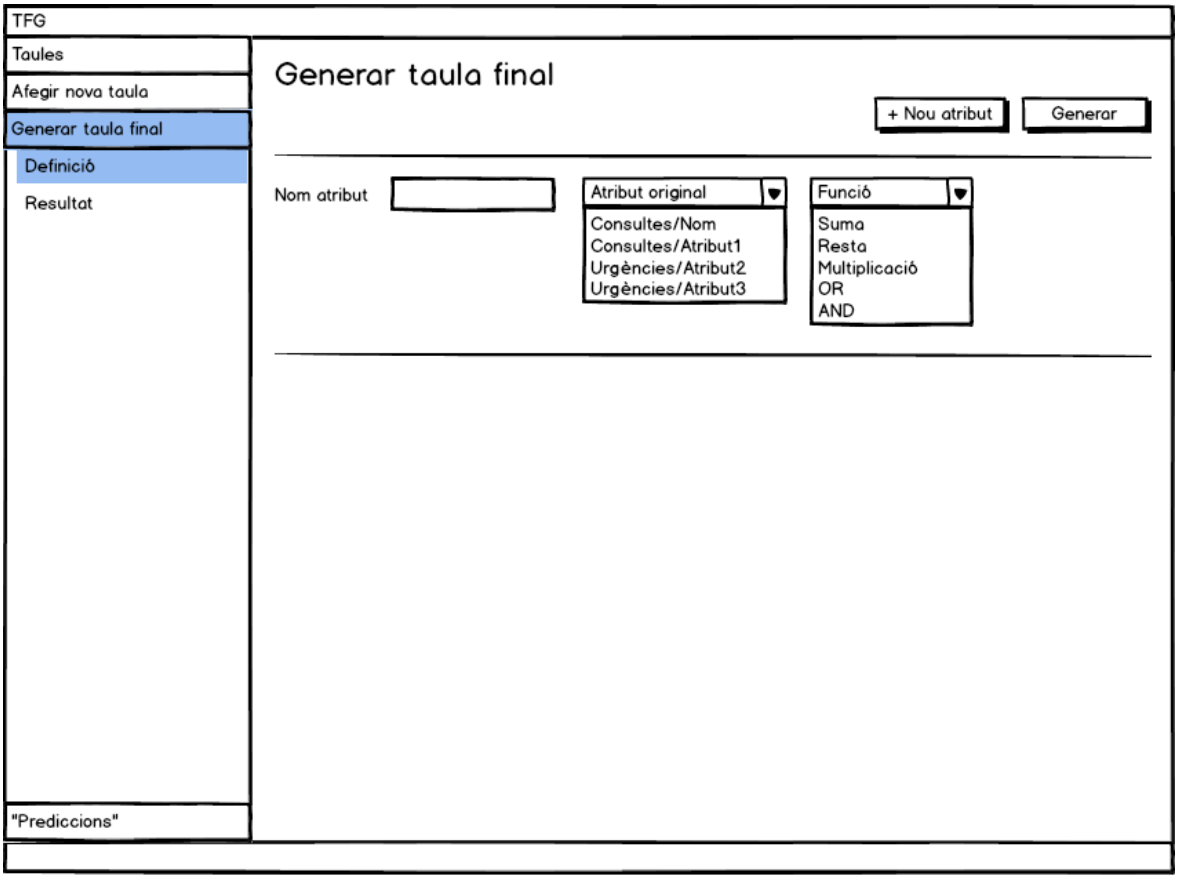
[13] Irina Rish. An empirical study of the naive bayes classifier. *RC 22230 (W0111-014)*, November 2001.

Apèndixs

Primers mockups




TFG
Taules
Consultes
Urgències
Urgències
Carregar dades
Esborrar taula
filtrar
ID
Nom
Edat
Atributs
1
Nom Cognoms
7
X
Afegir nova taula
Generar taula final
"Prediccions"
Created with Balsamiq - www.balsamiq.com


TFG
Taules
Afegir nova taula
Generar taula final
Definició
Resultat
"Prediccions"
Generar taula final
+ Nou atribut
Generar
Nom atribut
Atribut original
Consultes/Nom
Consultes/Atribut1
Urgències/Atribut2
Urgències/Atribut3
Funció
Suma
Resta
Multiplicació
OR
AND
Created with Balsamiq - www.balsamiq.com

Model per l'asma

Taula d'episodis

- cod_pacient - *int*
- cod_agenda - *string*
- cod_servei - *int*
- cod_episodi - *int* [**PK**]
- dataini - *string*
- tx - *string*
- obx - *string*
- val_obx - *string*
- data_naix - *string*
- cod_poblacio - *int*
- des_poblacio - *string*
- cod_provincia - *int*

Taula d'urgències

- codi - *int*
- pacient - *string* [**PK**]
- datafi - *string*
- ciediag_0 - *string*
- descdiag_0 - *string*
- servalta - *string*
- descripcion - *string*
- datanaix - *string*

Atributs de sortida

- edat - *numeric*
- codigopoblacion - *NUMERIC*
- frequenciatositdia - *NUMERIC*
- frequenciatositnit - *NUMERIC*
- costatrespirarnit - *NUMERIC*
- costatrespirardia - *NUMERIC*
- ingresatperasma - *NUMERIC*
- xiuletsnit - *NUMERIC*
- xiuletsdia - *NUMERIC*
- exercicioriure - *NUMERIC*
- urgenciesasma - *NUMERIC*
- altresmanifestacionsatopia - *0,1*
- edatinici - *NUMERIC*
- antecedentspersonals - *0,1*
- simptomesintercrisisasma - *0,1*
- canpunts - *NUMERIC*

- desencadenants - *0,1*
- moquetescatifes - *0,1*
- visitesurgencies - *0,1*
- freqcrisisasma - *NUMERIC*
- ingresosperasma - *0,1*
- tractamentbaseasma - *0,1*
- tractamentbaserinitis - *0,1*
- animalsdomicilihabitual - *0,1*
- faltesescolars - *0,1*
- simptomesambesforc - *0,1*
- crisisdasma - *0,1*
- immunoterapia - *0,1*
- dispneanocturna - *0,1*
- antecedentsinteres - *0,1*
- reaccionsite - *0,1*
- antecedentsatopia - *0,1*
- evoluciopatologia - *0,1*
- dispneadiurna - *0,1*
- estacionalitat - *0,1*
- respostabroncodilatadors - *0,1*
- medicaciorescat - *0,1*
- fumadorsdomicilihabitual - *0,1*
- simptomesrinoconjuntivitis - *0,1*
- tos-sibilancies - *0,1*
- asma - *0,1*
- rinitis - *0,1*
- acaros - *0,1*
- alergia - *0,1*
- rinoconjuntivitis - *0,1*
- pis - *0,1*
- humed - *0,1*
- casa - *0,1*
- ventilad - *0,1*
- solead - *0,1*
- peluch - *0,1*
- budesonida - *NUMERIC*
- fluticasona - *NUMERIC*
- ciclesonida - *NUMERIC*
- mometasona - *NUMERIC*
- montelukast - *NUMERIC*
- singulair - *NUMERIC*
- pluralais - *NUMERIC*
- salmeterolfluticasona - *NUMERIC*
- formoterolbudesonida - *NUMERIC*
- omalizumab - *NUMERIC*
- ocs - *NUMERIC*
- salbutamol - *NUMERIC*
- visitaUrgencies1Mes - *0,1*